

CTF 逆向题 shy

原创

KingZhang2000 于 2019-09-09 11:25:36 发布 1831 收藏 6

分类专栏: CTF 文章标签: CTF 逆向 upx脱壳

版权声明: 本文为博主原创文章, 遵循 CC 4.0 BY-SA 版权协议, 转载请附上原文出处链接和本声明。

本文链接: <https://blog.csdn.net/KingZhang2000/article/details/100654580>

版权



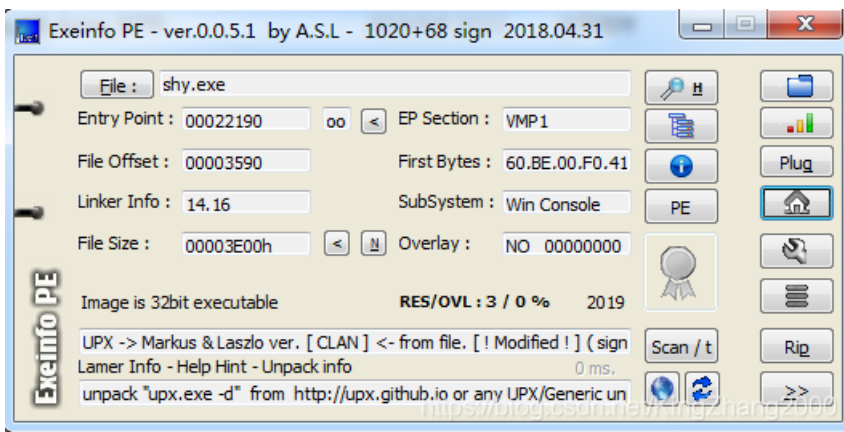
[CTF 专栏收录该内容](#)

2 篇文章 0 订阅

订阅专栏

这个题目是攻防大赛的逆向题, 是upx的壳, 由于当时手头的工具无法脱壳, 所以这题也就跳过了。经过学习了手工脱壳后, 感觉可以拿这个题练练手, 顺便写一个writeup。

首先查壳, 丢到ExeinfoPE里面看一下, 确定是upx壳

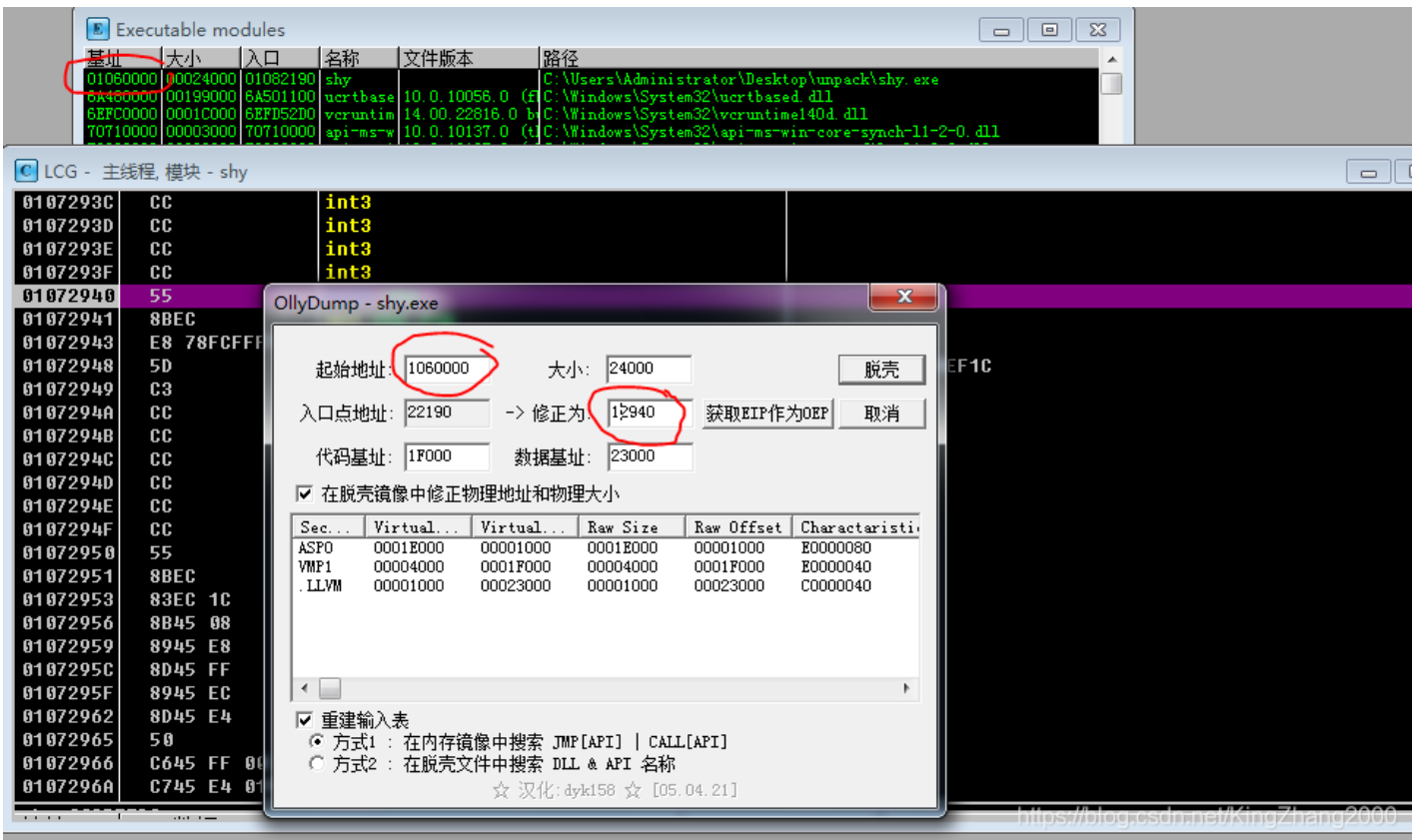


于是丢到OD里面进行脱壳处理, 由于是压缩壳, 跟踪起来比较麻烦, 我选择了个偷懒的办法, 下一个api访问断点

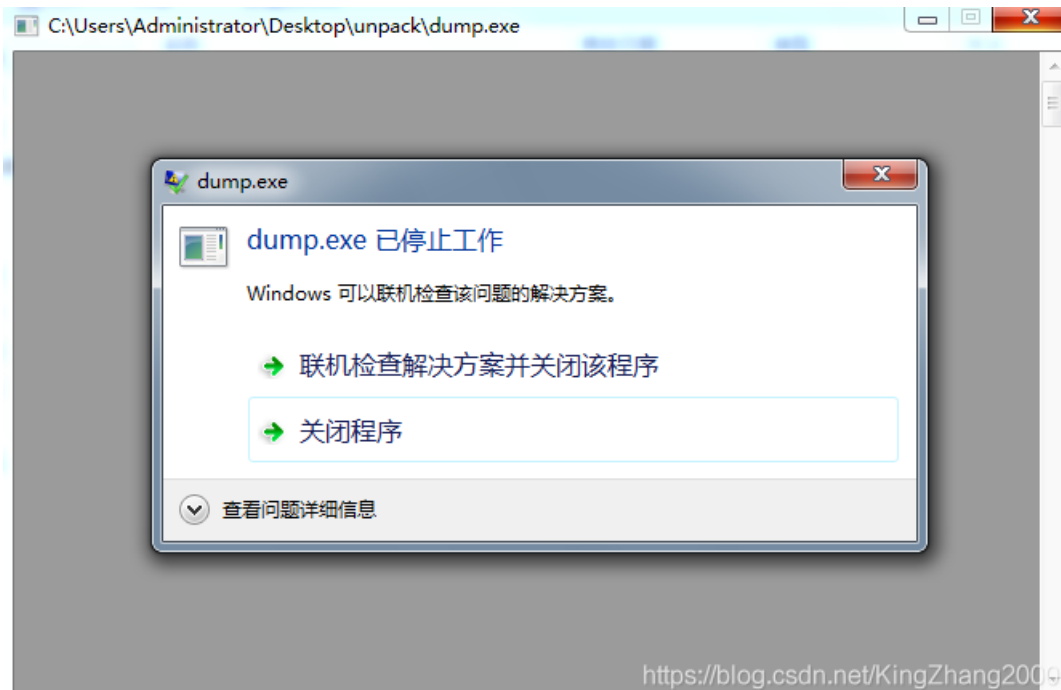
即: VirtualProtect, 运行3次F9后就跑飞了, 于是在2次运行后, 单步跟踪, 到OEP

0107293C	CC	int3	
0107293D	CC	int3	
0107293E	CC	int3	
0107293F	CC	int3	
01072940	55	push ebp	
01072941	8BEC	mov ebp,esp	
01072943	E8 78FCFFFF	call shy.010725C0	
01072948	5D	pop ebp	kerne132.771BEF1C
01072949	C3	retn	
0107294A	CC	int3	
0107294B	CC	int3	
0107294C	CC	int3	
0107294D	CC	int3	
0107294E	CC	int3	
0107294F	CC	int3	
01072950	55	push ebp	

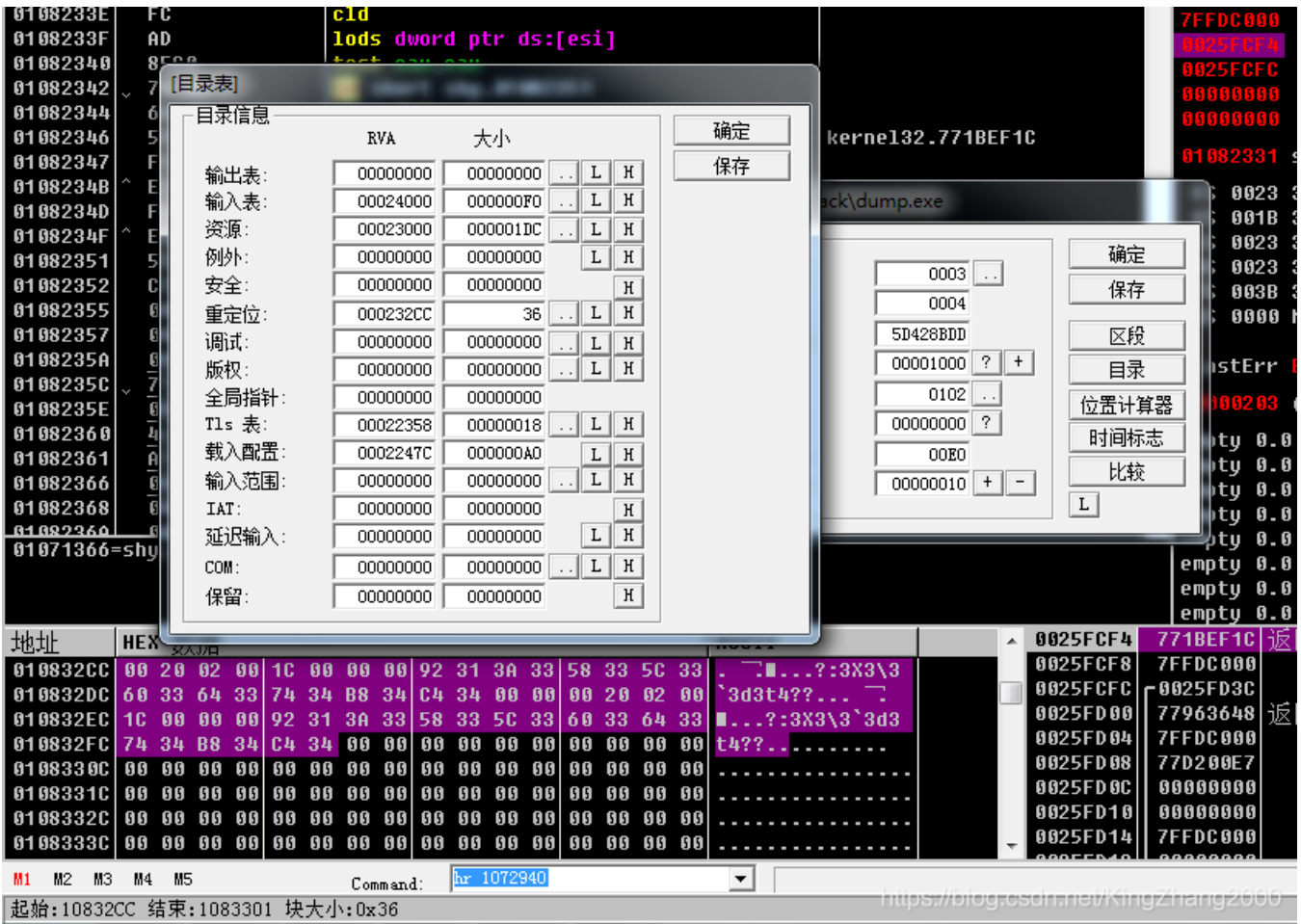
使用OD自带的插件进行脱壳, 注意基址和OEP的关系, 计算好后填入



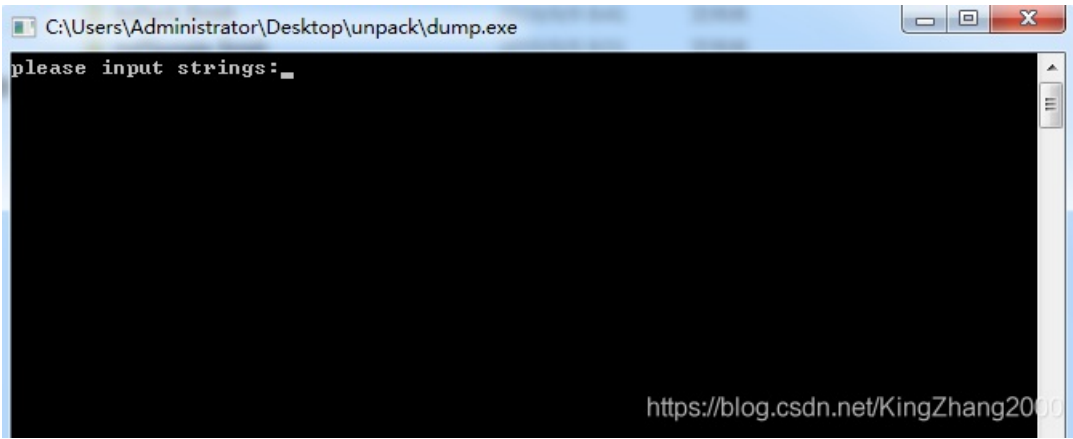
然后点击脱壳，双击发现不能运行。



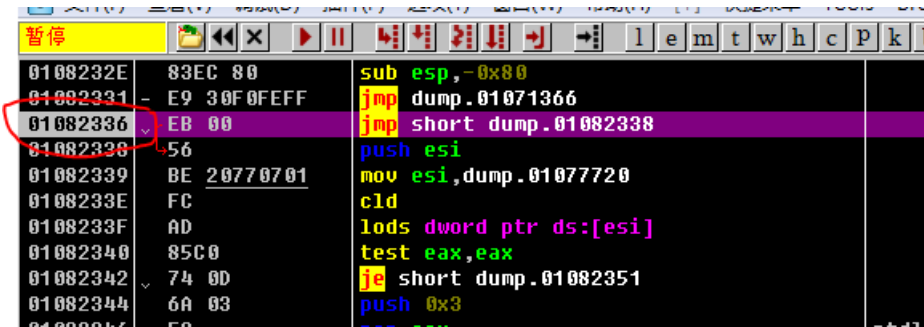
这个问题估计是重定向造成，于是修复一下重定向表的数据。



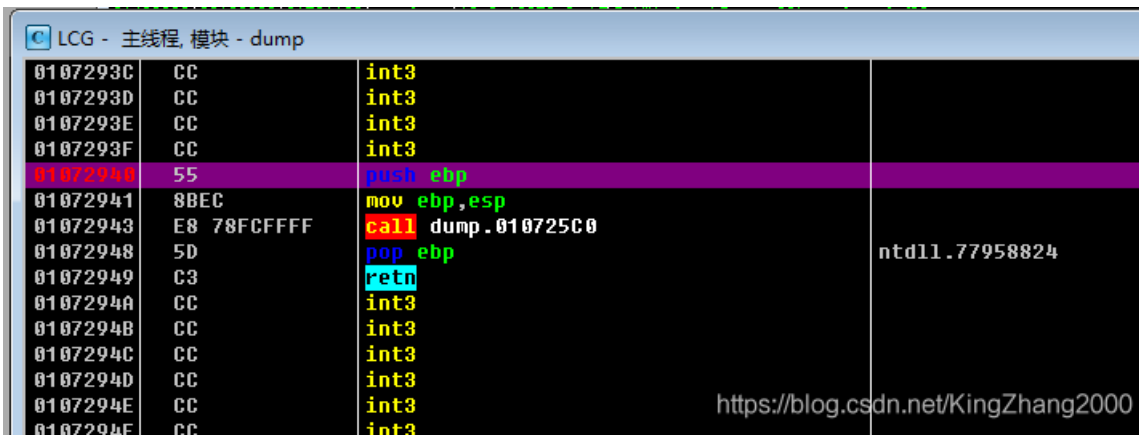
保存后，再次运行可以正常跑起来了



再次用OD载入，发现入口地址不是OEP



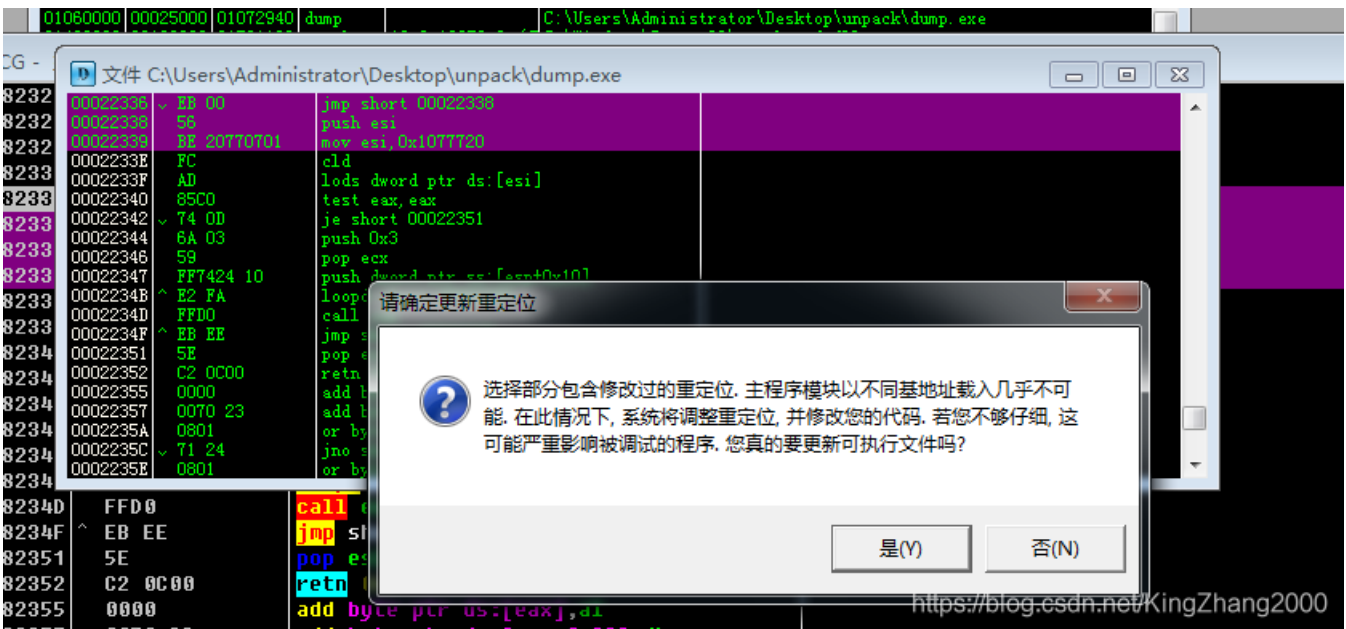
按说是已经解密完成了，于是我定位到OEP（0x1072940）一看究竟。



确实已经解密，那么为了方便调试，我直接用OD改一下入口代码就可以实现了。



然后把修改完毕的程序，重新保存到文件，由于有重定位会有下图的提示，点击是，然后右键保存一份 dump0.exe



继续，OD载入dump0.exe 发现修改成功，可以直接跳转到OEP行，然后单步跟踪，到输入后，发现后面的代码是个加密处理的代码，于是丢到IDA里面看一下这块对应的反编译代码。

```

37 sub_10710BE(Buf);
38 for ( i = 0; i <= 23; ++i )
39     v79[i] = *(&v29 + i) ^ *(&v53 + i) ^ Buf[i];
40 v4 = 54;
41 v5 = 108;
42 v6 = 106;
43 v7 = 104;
44 v8 = 44;
45 v9 = 33;
46 v10 = 59;
47 v11 = 58;
48 v12 = 43;
49 v13 = 38;
50 v14 = 112;
51 v15 = 37;
52 v16 = 105;
53 v17 = 42;
54 v18 = 97;
55 v19 = 61;
56 v20 = 83;
57 v21 = 99;
58 v22 = 52;
59 v23 = 35;
60 v24 = 112;
61 v25 = 116;
62 v26 = 42;
63 v27 = 37;
64 sub_107138B();
65 for ( j = 0; j < 24; ++j )
66 {
67     if ( *(&v4 + j) != v79[j] )
68     {
69         v77 = 1;
70         break;
71     }
72 }
73 if ( v77 == 1 )
74     sub_107104B("wrong");
75 else
76     sub_107104B("good");
77 sub_1071262(&savedregs, dword_1072028, 0, v0);
78 return sub_107123F((unsigned int)&savedregs, v20, v1);
79 }

```

buf 就是用户输入的字符串，if (*(&v4 + j) != v79[j]) 这个就是关键的比较，那么V79[]就应该是异或后的结果，也就是ben本题的密钥，于是在OD中定位值：6ljh,!;+&p%i*a=Sc4#pt*%

```

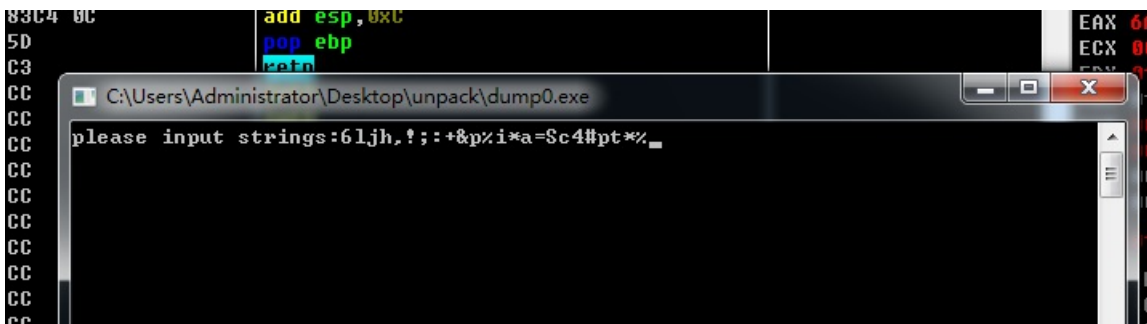
01071FA4 7D 2B jge short dump0.01071FD1
01071FA6 8B85 04FFFFFF mov eax,dword ptr ss:[ebp-0xFC]
01071FAC 0FBE8C05 10FFFFFF movsx ecx,byte ptr ss:[ebp+eax-0xF0]
01071FB4 8B95 04FFFFFF mov edx,dword ptr ss:[ebp-0xFC]
01071FBA 0FBE4415 C4 movsx eax,byte ptr ss:[ebp+edx-0x3C]
01071FBF 3BC8 cmp ecx,eax
01071FC1 74 0C jc short dump0.01071FCF
01071FC3 C785 7CFFFFFF 01000 mov dword ptr ss:[ebp-0x84],0x1
01071FCD EB 02 jmp short dump0.01071FD1
01071FCF EB BD jmp short dump0.01071F8E
01071FD1 83BD 7CFFFFFF 01 cmp dword ptr ss:[ebp-0x84],0x1
01071FD8 75 0F jnz short dump0.01071FE9
01071FDA 68 5C7B0701 push dump0.01077B5C ASCII "wrong"
01071FDF E8 67F0FFFF call dump0.0107104B
01071FE4 83C4 04 add esp,0x4
01071FE7 EB 0D jmp short dump0.01071FF6
01071FE9 68 387C0701 push dump0.01077C38 ASCII "good"
01071FEE E8 58F0FFFF call dump0.0107104B
01071FF3 83C4 04 add esp,0x4
01071FF6 33C0 xor eax,eax
01071FF8 52 push edx
01071FF9 8BCD mov ecx,ebp
01071FEB 50 push eax

```

堆栈 ss:[0012F88C]=36 ('6')
ecx=44BD78C7

地址	HEX 数据	ASCII
0012F88C	36 6C 6A 68 2C 21 3B 3A 2B 26 70 25 69 2A 61 3D	61jh,!;:+&p%i*a=
0012F89C	53 63 34 23 70 74 2A 25 CC CC CC CC CC CC CC CC	Sc4#pt*% 汤汤汤汤
0012F8AC	18 00 00 00 CC CC CC CC CC CC CC CC 31 7A 73 77	汤汤汤汤 1z5w
0012F8BC	34 33 38 6F 4F 46 75 35 69 34 6E 64 30 66 5F 63	438o0Fu5iund0f c
0012F8CC	48 32 7A 31 CC CC CC CC CC CC CC CC 61 7A 78 78	H2z 汤汤汤汤 azxx

接下来就简单了，直接把这个密钥输入，然后再次定位到这块就能得到flag了



```

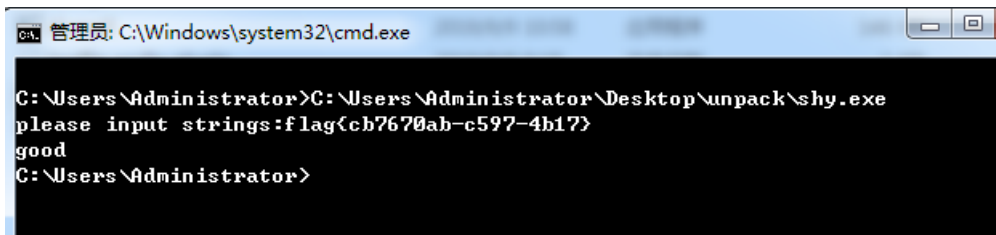
01071F97 8985 04FFFFFF mov dword ptr ss:[ebp-0xFC],eax
01071F9D 83BD 04FFFFFF 18 cmp dword ptr ss:[ebp-0xFC],0x18
01071FA4 7D 2B jge short dump0.01071FD1
01071FA6 8B85 04FFFFFF mov eax,dword ptr ss:[ebp-0xFC]
01071FAC 0FBE8C05 10FFFFFF movsx ecx,byte ptr ss:[ebp+eax-0xF0]
01071FB4 8B95 04FFFFFF mov edx,dword ptr ss:[ebp-0xFC]
01071FBA 0FBE4415 C4 movsx eax,byte ptr ss:[ebp+edx-0x3C]
01071FBF 3BC8 cmp ecx,eax
01071FC1 74 0C jc short dump0.01071FCF
01071FC3 C785 7CFFFFFF 01000 mov dword ptr ss:[ebp-0x84],0x1
01071FCD EB 02 jmp short dump0.01071FD1
01071FCF EB BD jmp short dump0.01071F8E
01071FD1 83BD 7CFFFFFF 01 cmp dword ptr ss:[ebp-0x84],0x1
01071FD8 75 0F jnz short dump0.01071FE9
01071FDA 68 5C7B0701 push dump0.01077B5C ASCII "wrong"
01071FDF E8 67F0FFFF call dump0.0107104B
01071FE4 83C4 04 add esp,0x4

```

堆栈 ss:[0012F940]=66 ('f')
eax=00000066

地址	HEX 数据	ASCII
0012F940	66 6C 61 67 7B 63 62 37 36 37 30 61 62 2D 63 35	flag{cb7670ab-c5
0012F950	39 37 2D 34 62 31 37 7D CC CC CC CC CC CC CC CC	97-4b17} 汤汤汤汤
0012F960	CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC CC	汤汤汤汤汤汤汤汤
0012F970	CC CC CC CC CC CC CC CC C4 20 7C 7C 90 F9 12 00	汤汤汤汤? 恩.
0012F980	CE 28 07 01 01 00 00 00 20 A9 28 00 38 AE 28 00	?#...?.8?.
0012F990	EC F9 12 00 37 27 07 01 54 20 7C 7C 30 19 26 00	襖.7'■丹 0&.
0012F9A0	10 FA 12 00 36 23 08 01 CC 5C 94 77 EA 6B 98 77	■?6#■ 蘇搭坏梅
0012F9B0	CC F9 12 00 B8 F9 12 00 80 96 98 00 00 00 00 00	始.根.料....

最终得到flag: flag{cb7670ab-c597-4b17} 输入到shy.exe 验证一下 :)



```
管理员: C:\Windows\system32\cmd.exe
C:\Users\Administrator>C:\Users\Administrator\Desktop\unpack\shy.exe
please input strings:flag{cb7670ab-c597-4b17}
good
C:\Users\Administrator>
```